B1

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 113 617 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
04.07.2001 Bulletin 2001/27

(51) Int Cl.^{7}: H04L 9/30

(21) Application number: 00121783.5

(22) Date of filing: 05.10.2000

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU
MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 21.12.1999 US 468703

(71) Applicant: ContentGuard Holdings, Inc.
Wilmington, Delaware 19803 (US)

(72) Inventor: Wang, Xin
Los Angeles, CA 90007 (US)

(74) Representative: Grünecker, Kinkeldey,
Stockmair & Schwanhäusser Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

(54) System and method for transferring the right to decode messages

(57) Methods for transferring among key holders in encoding and cryptographic systems the right to decode and decrypt messages in a way that does not explicitly reveal decoding and decrypting keys used and the original messages. Such methods are more secure and more efficient than typical re-encoding and re-encryption schemes, and are useful in developing such applications as document distribution and long-term file protection.

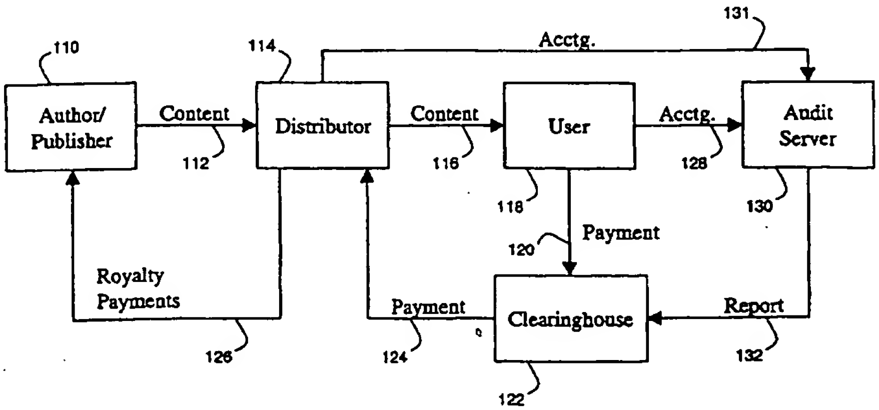

Fig. 1

Printed by Jouve, 75001 PARIS (FR)

Description

FIELD OF THE INVENTION

[0001] The invention relates to cryptographic methods, and more particularly to systems and methods for delegating decryption rights without revealing private keys or message contents.

BACKGROUND OF THE INVENTION

[0002] One of the most important issues impeding the widespread distribution of digital documents via electronic commerce is the current lack of protection of the intellectual property rights of content owners during the distribution and use of those digital documents. Efforts to resolve this problem have been termed "Intellectual Property Rights Management" ("IPRM"), "Digital Property Rights Management" ("DPRM"), "Intellectual Property Management" ("IPM"), "Rights Management" ("RM"), and "Electronic Copyright Management" ("ECM").

[0003] A document, as the term is used herein, is any unit of information subject to distribution or transfer, including but not limited to correspondence, books, magazines, journals, newspapers, other papers, software, photographs and other images, audio and video clips, and other multimedia presentations. A document may be embodied in printed form on paper, as digital data on a storage medium, or in any other known manner on a variety of media.

[0004] In the world of printed documents, a work created by an author is usually provided to a publisher, which formats and prints numerous copies of the work. The copies are then sent by a distributor to bookstores or other retail outlets, from which the copies are purchased by end users.

[0005] While the low quality of copying and the high cost of distributing printed material have served as deterrents to the illegally copying of most printed documents, it is far too easy to copy, modify, and redistribute unprotected electronic documents. Accordingly, some method of protecting electronic documents is necessary to make it harder to illegally copy them. This will serve as a deterrent to copying, even if it is still possible, for example, to make hardcopies of printed documents and duplicate them the old-fashioned way.

[0006] With printed documents, there is an additional step of digitizing the document before it can be redistributed electronically; this serves as a deterrent. Unfortunately, it has been widely recognized that there is no viable way to prevent people from making unauthorized distributions of electronic documents within current general-purpose computing and communications systems such as personal computers, workstations, and other devices connected over local area networks (LANs), intranets, and the Internet. Many attempts to provide hardware-based solutions to prevent unauthorized copying have proven to be unsuccessful.

[0007] Two basic schemes have been employed to attempt to solve the document protection problem: secure containers and trusted systems.

[0008] A "secure container" (or simply an encrypted document) offers a way to keep document contents encrypted until a set of authorization conditions are met and some copyright terms are honored (e.g., payment for use). After the various conditions and terms are verified with the document provider, the document is released to the user in clear form. Commercial products such as IBM's Cryptolopes and InterTrust's Digiboxes fall into this category. Clearly, the secure container approach provides a solution to protecting the document during delivery over insecure channels, but does not provide any mechanism to prevent legitimate users from obtaining the clear document and then using and redistributing it in violation of content owners' intellectual property.

[0009] Cryptographic mechanisms are typically used to encrypt (or "encipher") documents that are then distributed and stored publicly, and ultimately privately deciphered by authorized users. This provides a basic form of protection during document delivery from a document distributor to an intended user over a public network, as well as during document storage on an insecure medium.

[0010] In the "trusted system" approach, the entire system is responsible for preventing unauthorized use and distribution of the document. Building a trusted system usually entails introducing new hardware such as a secure processor, secure storage and secure rendering devices. This also requires that all software applications that run on trusted systems be certified to be trusted. While building tamper-proof trusted systems is still a real challenge to existing technologies, current market trends suggest that open and untrusted systems such as PC's and workstations will be the dominant systems used to access copyrighted documents. In this sense, existing computing environments such as PC' s and workstations equipped with popular operating systems (e.g., Windows and UNIX) and render applications (e.g., Microsoft Word) are not trusted systems and cannot be made trusted without significantly altering their architectures.

[0011] Accordingly, although certain trusted components can be deployed, one must continue to rely upon various unknown and untrusted elements and systems. On such systems, even if they are expected to be secure, unanticipated bugs and weaknesses are frequently found and exploited.

[0012] One particular issue arises in the context of document distribution, as described generally above. In the traditional model of document distribution, the content author and the publisher typically do not handle distribution; a separate party with distribution expertise is given that responsibility. Furthermore, while it is possible to encrypt a document (using standard techniques) so that multiple recipients can decrypt it, it is not usually known at the time a work is created who the ultimate users will be. It makes more sense for the distributor to determine who the end users will be, and to distribute the document to them as desired. If, as in traditional model, the original work of authorship is sent to a publisher and a distributor in the clear, that is a point of vulnerability for the work.

[0013] A similar problem arises in office settings, for example, in which it is frequently desirable to designate what is variously called a document agent, surrogate, or delegate. In this situation, it is often useful to be able to give an administrative assistant or secretary the right to decrypt certain document not intended directly for that person.

[0014] Considering the problem more broadly, in a networked environment, messages are often passed to recipients other than their initially intended ones. When message confidentiality is a concern and encrypted messages are forwarded, it is very desirable to allow one to decrypt these messages on behalf of another. To be concrete, suppose that Bob is the one who needs to read some message that is initially encrypted for Alice. One trivial solution is that Alice simply reveals her decryption key to Bob so that Bob can use it to decrypt the message himself. This requires Alice to trust Bob totally, which may not be acceptable to Alice. Another way to accomplish this task is to let Alice first decrypt the message, then re-encrypt it for Bob and finally send the newly encrypted message to Bob so that he can decrypt. Though the message is communicated securely, this solution is less efficient as it requires two decryption and one encryption operations in order for Bob to obtain the message. More importantly, in some situations such re-encryption solution is not even applicable or desirable. For example, Alice may not have access to the encrypted message, as it may be sent by its originator directly to Bob for communication efficiency and other considerations. Also, decrypting the encrypted message to a clear version, even if only for a short time, can be a substantial vulnerability.

[0015] Accordingly, it would be desirable to have an encryption/decryption framework that supports the ability to transfer the right to decode messages. Such a framework would allow a delegate to, essentially, authorize the re-encryption of a message for another party's use without first decrypting the original message. It would also be useful for this to be possible without the delegate ever having possession of the encrypted message.

SUMMARY OF THE INVENTION

[0016] How to transfer the right to decrypt from one key holder to another in a secure and efficient way is the subject of proxy encryption. Some specific proxy encryption schemes have been recently proposed to convert messages encrypted for one key into messages encrypted for another without revealing secret decryption keys and original messages to the public. Mambo and Okamoto have introduced several private, non-commutative, message-independent proxy encryption schemes. Blaze and Strauss have introduced a public, commutative, message-independent proxy encryption scheme.

[0017] In this disclosure, the same general problem is initially addressed but in the more general context of encoding schemes. Encoding schemes considered in this disclosure differ from encryption schemes or cryptosystems in that they do not necessarily have any security-related requirements. For an encoding scheme to be an encryption scheme, it is necessary that an eavesdropper, upon seeing an encoded message, should be unable to determine either the original message or the key used to decode the message. Working with encoding schemes makes it possible to build applications with lightweight security but high implementation efficiency, such as efficient massive document distribution and updating of ciphertext with new keys to protect long-term encrypted messages. In this disclosure, a class of encoding schemes is defined, and several example schemes are given. A process by which new schemes can be constructed using existing ones is also offered herein.

[0018] Several more formal proxy encryption schemes are then presented. A proxy encryption scheme is an encryption scheme that allows a designated key holder to decrypt messages on behalf of another key holder. This disclosure introduces two new proxy encryption schemes based on the known ElGamal scheme, with improved functionalities over existing proxy encryption schemes. They are public in the sense that proxy-related information and transformations can be safely made to the public, and at the same time non-commutative in terms of trust relationships among involved key holders. Applications of these new schemes to massive document distribution and file protection are also presented.

[0019] The basic idea in the methods present in this disclosure is as follows: in order for Alice to transfer the right to decode to Bob, Alice generates a transfer key t for Bob. With the transfer key t, Bob can re-encrypt the message initially encoded for Alice and subsequently decrypt it using his own key. Much like in proxy encryption, the transfer is performed in such a way that the transfer key does not explicitly reveal the decoding keys of either Alice or Bob, or the original message.

[0020] How to delegate the right to decrypt from one key holder to another in secure and efficient ways is the subject of proxy encryption. Very recently, some specific proxy encryption schemes have been proposed to convert messages encrypted for one key into messages encrypted for another without revealing secret decryption keys and original messages to the public. Mambo and Okamoto have described three proxy encryption schemes for the ElGamal and RSA encryption schemes. M. Mambo and E. Okamoto, "Proxy cryptosystems: Delegation of the power to decrypt ciphertexts, " *IEICE Trans. on Fundamentals*, Vol. E80-A, No. 1, pp. 54-63 (1997). For the situation mentioned above, their schemes have better computational performance over the re-encryption scheme, but for security reasons require the presence of the original key holder Alice in the message conversion. Moreover, the schemes themselves do not help specifying who is the key holder that Alice wants to delegate the decryption right to. The scheme proposed by Blaze and Strauss, on the other hand, does not have these shortcomings. It is a modification of the ElGamal encryption scheme. M. Blaze and M. Strauss, "Proxy Cryptography," Draft, AT&T Research Labs, ftp://ftp.research.att.com/dist/mab/proxy.ps (May 1997). One very appearing feature of the Blaze and Strauss scheme is that it permits communicating proxy related information and performing the message conversion in public. But it introduces a more serious problem: it is commutative in the sense that Bob is able to obtain Alice's decryption key. This type of commutativity makes the proxy encryption scheme obsolete, as the entire scheme can be well simplified to giving Alice's key to Bob and letting Bob decrypt. Another issue (not necessarily a problem) created by this scheme is that once Bob has been granted the decryption right by Alice, he can decrypt all messages that are originally for Alice. This message-independence may be useful in some cases such as self-delegation but is not be desirable in many practical applications where the original key holder wants to be selective on which messages the delegated decryption is allowed.

[0021] Accordingly, the proxy encryption schemes according to the present invention, which are public and non-commutative, eliminate some of the disadvantages of other known cryptosystems.

[0022] In this disclosure, two new proxy encryption schemes are then introduced. They are all based on the ElGamal public-key encryption scheme and have comparable computational performance. Essentially, they have retained the following desirable features of the existing schemes: (i) public: the presence of the original key holder is not required after proxy information is generated, and proxy related information and operations can communicated and conducted in public; (ii) non-commutative: key holders do not have to trust each other in regard to their private decryption keys; and (iii) restricted: the key holder to whom the decryption right is delegated to is specified, and the proxy information (key) is message dependent.

[0023] Finally, delegating the right to decrypt messages is then described in the context of the Cramer-Shoup cryptosystem, which bears some advantages over other systems.

[0024] In a further embodiment the method comprises the step of providing the transformed message to the recipient.

[0025] In a further embodiment the method comprises the step of decrypting the transformed message using information selected from the private key corresponding to the recipient and any available public information.

[0026] In a further embodiment the method comprises the step of decrypting the transformed message using the private key corresponding to the recipient.

[0027] In a further embodiment the encrypted message comprises a first portion and a second portion, the first portion encoding a generator and a random key, and the second portion encoding the original message, the public key corresponding to the grantor, and the random key.

[0028] In a further embodiment the applying step operates on the second portion of the encrypted message.

[0029] In a further embodiment the encrypted message comprises a first portion and a second portion, the first portion encoding the original message, a generator, and a random key, and the second portion encoding the public key corresponding to the grantor and the random key.

[0030] In a further embodiment the applying step operates on the second portion of the encrypted message.

[0031] In a further embodiment the encrypted message comprises a first portion and a second portion, the first portion encoding the original message, a generator, and a random key, and the second portion encoding the public key corresponding to the grantor and the random key.

[0032] In a further embodiment the applying step operates on the second portion of the encrypted message.

[0033] In a further embodiment the original message is passed to a recipient through at least one additional intermediate grantor by repeating the generating and applying steps for each additional intermediate grantor.

[0034] These and other features and advantages of the present invention are apparent from the Figures as fully described in the Detailed Description of the Invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0035]

FIGURE 1 is a block diagram of an electronic document distribution system capable of operation according to the invention;

FIGURE 2 is a block diagram illustrating the encoding operations performed when delegating the authority to decrypt a message in a method according to the invention;

FIGURE 3 is a flow chart illustrating the general steps performed in transforming an encoded message for decoding by another;

FIGURE 4 is a block diagram schematically illustrating the parties involved in a system adapted for the delegation of the authority to decrypt messages;
FIGURE 5 is a flow chart illustrating the steps performed in a generic proxy encryption scheme;
FIGURE 6 is a flow chart illustrating the steps performed in encrypting and decrypting a message according to the ElGamal cryptosystem;
FIGURE 7 is a flow chart illustrating the steps performed in a known ElGamal-based proxy encryption and decryption scheme proposed by Mambo and Okamoto;
FIGURE 8 is a flow chart illustrating the steps performed in a known ElGamal-based proxy encryption and decryption scheme proposed by Blaze and Strauss;
FIGURE 9 is a flow chart illustrating the steps performed in a first embodiment of an ElGamal-based proxy encryption and decryption scheme according to the invention;
FIGURE 10 is a flow chart illustrating the steps performed in a second embodiment of an ElGamal-based proxy encryption and decryption scheme according to the invention;
FIGURE 11 is a flow chart illustrating the steps performed in a document distribution scheme according to the invention;
FIGURE 12 is a flow chart illustrating the steps performed in a file protection scheme according to the invention;
FIGURE 13 is a flow chart illustrating the steps performed in encrypting and decrypting a message according to the Cramer-Shoup cryptosystem; and
FIGURE 14 is a flow chart illustrating the steps performed in an embodiment of a Cramer-Shoup-based proxy encryption and decryption scheme according to the invention.
The Figures are more fully explained in the following Detailed Description of the Invention.

DETAILED DESCRIPTION OF THE INVENTION

[0036] Figure 1 represents a top-level functional model for a system for the electronic distribution of documents, which as defined above, may include correspondence, books, magazines, journals, newspapers, other papers, software, audio and video clips, and other multimedia presentations.

[0037] An author (or publisher) 110 creates a document's original content 112 and passes it to a distributor 114 for distribution. Although it is contemplated that the author may also distribute documents directly, without involving another party as a publisher, the division of labor set forth in Figure 1 is more efficient, as it allows the author/publisher 110 to concentrate on content creation, and not the mechanical and mundane functions taken over by the distributor 114. Moreover, such a breakdown would allow the distributor 114 to realize economies of scale by associating with a number of authors and publishers (including the illustrated author/publisher 110).

[0038] The distributor 114 then passes modified content 116 to a user 118. In a typical electronic distribution model, the modified content 116 represents an reencrypted version of the original encrypted content 112; the distributor 114 first decrypts the original content 112 and then re-encrypts it with the user 118's public key; that modified content 116 is customized solely for the single user 118. The user 118 is then able to use his private key to decrypt the modified content 116 and view the original content 112.

[0039] A payment 120 for the content 112 is passed from the user 118 to the distributor 114 by way of a clearinghouse 122. The clearinghouse 122 collects requests from the user 118 and from other users who wish to view a particular document. The clearinghouse 122 also collects payment information, such as debit transactions, credit card transactions, or other known electronic payment schemes, and forwards the collected users' payments as a payment batch 124 to the distributor 114. Of course, it is expected that the clearinghouse 122 will retain a share of the user's payment 120. In turn, the distributor 114 retains a portion of the payment batch 124 and forwards a payment 126 (including royalties) to the author and publisher 110. In one embodiment of this scheme, the distributor 114 awaits a bundle of user requests for a single document before sending anything out. When this is done, a single document with modified content 116 can be generated for decryption by all of the requesting users. This technique is well-known in the art.

[0040] In the meantime, each time the user 118 requests (or uses) a document, an accounting message 128 is sent to an audit server 130. The audit server 130 ensures that each request by the user 118 matches with a document sent by the distributor 114; accounting information 131 is received by the audit server 130 directly from the distributor 114. Any inconsistencies are transmitted via a report 132 to the clearinghouse 122, which can then adjust the payment batches 124 made to the distributor 114. This accounting scheme is present to reduce the possibility of fraud in this electronic document distribution model, as well as to handle any time-dependent usage permissions that may result in charges that vary, depending on the duration or other extent of use.

[0041] The foregoing model for electronic commerce in documents, shown in Figure 1, is in common use today. As will be shown in detail below, it is equally applicable to the system and method set forth herein for the distribution of self-protecting documents.

Proxy Encoding Schemes

[0042] For simplicity, initially consider encoding schemes of the following type. An encoding system consists of four components: (i) a message space X which is a collection of possible messages, (ii) a key space K which is a set of possible keys, (iii) a computationally efficient encoding transformation $E : K \times X \rightarrow X$ and (iv) a computationally efficient decoding transformation $D: K \times X \rightarrow X$. For each $k \in K$, the encoding transformation $E_k : X \rightarrow X$ and decoding transformation $D_k : X \rightarrow X$ are injection (one-to-one) mappings on X, and they satisfy that, for every message $x \in X$,

$$D_k(E_k(x)) = x.$$

Certainly, such defined encoding schemes can be varied in several ways to cover a wider range of ones. One is to differentiate the space of encoded messages from the one of original messages, and another is to consider that keys used for encoding and decoding are different. In terms of cryptography, the encoding schemes considered below are essentially private-key (or, more precisely, symmetric), endomorphic cryptosystems.

[0043] Such defined encoding schemes have some advantageous properties. Given an encoding scheme (X, K, E, D), each encoding transformation and its corresponding decoding transformation are inverse transformation of each other; that is, for each $k \in K$,

$$D_k = (E_k)^{-1} \text{ and } E_k = (D_k)^{-1}.$$

If X is a finite set, each encoding or decoding transformation is just a permutation on X.

[0044] Classic, symmetric-key encryption schemes are encoding schemes. Here are some of them.

[0045] **XOR Scheme *X*.** In this scheme, the message space X is the set $\mathbf{B}_n$ of all n-bit binary strings for some integer $n > 0$, and so is the key space K. The number of possible messages and the number of possible keys are both 2^n. For each message x and each key k, the encoding is

$$y = E_k(x) = x \oplus k$$

and the decoding of message y is

$$x = D_k(y) = y \oplus k;$$

where $\oplus$ represents the bit-wise XOR (exclusive or) operation.

[0046] **Multiplicative Scheme *M*.** A message in this scheme is an element in $X = \mathbf{Z}_n = \{0, 1, ..., n\text{-}1\}$ for some integer $n > 0$. A key is also an element a in $\mathbf{Z}_n$ but satisfying gcd(a, n) = 1, where the "gcd" function specifies the greatest common integer divisor of the two arguments. That is, the key space K consists of the elements in the multiplicative group $\mathbf{Z}_n^* = \{a \in \mathbf{Z}_n | \gcd(a,n)=1\}$. The encoding of a message x with a key a is

$$y = E_a(x) = ax(\text{mod} n)$$

and the decoding of a message y with a key a is

$$x = D_a(y) = a^{-1}y(\text{mod} n),$$

where a^{-1} is the multiplicative inverse of a modulo n; that is, a^{-1} is an element in $\mathbf{Z}_n$ such that $aa^{-1}(\text{mod } n) = a^{-1}a(\text{mod } n) = 1$. Note that the condition on a, gcd(a, n) = 1, is used to guarantee that a has the inverse a^{-1}. It is known that the number of such as is equal to the value of the Euler phi-function

$$\phi(n) = \prod_{i=1}^{m} (p_i^{e_i} - p_i^{e_i-1})$$

where

$$n = \prod_{i=1}^{m} p_i^{e_i}$$

is the prime decomposition of n. So the number of keys in the scheme M is $\o(n)$.

[0047] Shift Scheme *S*. Messages and keys of the shift scheme are all elements in $\mathbf{Z}_n = \{0, 1, \ldots, n\text{-}1\}$ for some integer $n > 0$; that is, $X = K = Z_n$. Thus, the number of messages and the number of keys in the shift scheme are all equal to n. To encode a message x with a key b, one calculates

$$y = E_b(x) = x + b \pmod{n}$$

and to decode a message y with b, one computes

$$x = D_b(y) = y - b \pmod{n}.$$

[0048] Substitution Scheme *P*. This scheme is also defined over $X = \mathbf{Z}_n$. However, the key space $K = \Pi_n$ consists of all permutations of elements in $\mathbf{Z}_n$. Thus, the total number of keys is $n!$. For each permutation $p \in \Pi_n$, the encoding is

$$y = E_p(x) = p(x),$$

while the decoding is

$$x = D_p(y) = p^{-1}(y),$$

where p^{-1} is the inverse permutation of p.

[0049] It should be noted that the multiplicative and shift schemes are special cases of the substitution scheme which include only $\o(n)$ and n of the $n!$ possible permutations of n elements, respectively.

[0050] New encoding schemes can be constructed by combining existing ones. One way is to form their "product." Suppose S and S' are two encoding schemes with the same message space X. The product of S and S', denoted by $S \times S'$, has the same message space X. A key of the product scheme has the form (k, k'), where k and k' are keys of S and S', respectively. The encoding and decoding transformations of the product scheme are defined as follows: for each key $(k, k') \in K$,

$$E_{(k,k')}(x) = E_{k'}(E_k(x))$$

and

$$D_{(k,k')}(x) = D_k(D'_{k'}(c)).$$

That is, the message x is first encoded with E_k, and the resulting message is then "re-encoded" with $E_{k'}$. Decoding is similar, but it is done in the reverse order.

[0051] It is straightforward to check that the product construction is always associative: $(S \times S') \times S'' = S \times (S' \times S'')$. If an encoding scheme S is taken to form the product with itself, one obtains the scheme $S \times S$, denoted by S^2. If the n-fold product is taken, the resulting scheme, denoted by S^n, is called an iterated encoding scheme.

[0052] A simple example to illustrate the definition of product encoding schemes is as follows.

[0053] Affine Scheme *A*. This scheme is also defined over $X = \mathbf{Z}_n$. A key of the affine scheme is a pair of integers (a, b) in $\mathbf{Z}_n$, where $\gcd(a, n) = 1$. The encoding transformation is

$$y = E_{(a,b)}(x) = (ax + b) \pmod{n}$$

and the decoding transformation is

$$x = D_{(a,b)}(y) = a^{-1}(y - b) \pmod{n}$$

where a^{-1} is the modular inverse of a modulo n. These transformations of the type $ax + b$ are usually called affine transformations, hence the name affine scheme. Note that the scheme A reduces to the multiplicative scheme M when $b = 0$ and the shift scheme S when $a = 1$. Thus, M and S are special cases of A. On the other hand, A is their product $M \times S$. As seen before, a key in the multiplicative scheme M is an element $\alpha \in \mathbf{Z}_n^*$; the corresponding encoding transformation is $E_a(x) = ax \pmod{n}$. A key in the shift scheme is an element $b \in \mathbf{Z}_n$, and the corresponding encoding transformation is $E_b(x) = x + b \pmod{n}$. Hence, a key in the product scheme $\mathrm{M} \times S$ has the form $(a,b) \in \mathbf{Z}_n^* \times \mathbf{Z}_n$, and its encoding is

$$E_{(a,b)}(x) = E_b(E_a(x)) = ax + b \pmod{n}.$$

This is precisely the definition of the encoding transformation in the affine scheme. Similarly, the decoding transformation in the affine scheme is the composition of the decoding transformations of the shift and multiplicative schemes.

[0054] The objective of transferring the right to decode messages in any given encoding scheme (X, K, E, D) can be stated as follows: for any given message $x \in X$ and keys $k,k' \in K$, convert in some efficient way the encoded message $y = E_k(x)$ using the key k into the encoded message $y' = E_{k'}(x)$ using the key K' so that the new message y' can be decoded correctly using the key k'. If this can be achieved, it is said that the right to decode the message y has been transferred or delegated from the key holder of k to the key holder of k'.

[0055] Figure 2 illustrates the transformation π 210 that is needed to achieve the objective. The thick lines 212, 214, and 216 representing transformations E_k, π, and $D_{k'}$, respectively, form a sequence of steps that encodes a message x with one key k, converts the encoded message into the other one encoded with another key K', and decodes the message using the key k'. The thin lines 218 and 220, representing the transformations $E_{k'}$ and D_k, respectively, show other possible encoding and decoding operations that may be performed.

[0056] In many cases, the key space K of an encoding scheme is not merely a set. Equipped with some operation "·", K may possess some mathematical structure. For instance, the key spaces of all the example schemes given in the previous section can be equipped with some operations to become mathematical groups. Table 1, below, shows some of these operations, where stands for the composition operator of permutations and

$$\cdot : (\mathrm{Z}_n^* \times \mathrm{Z}_n) \times (\mathrm{Z}_n^* \times \mathrm{Z}_n) \rightarrow \mathrm{Z}_n^* \times \mathrm{Z}_n$$

is defined as

$$(a,b) \cdot (a',b') = (a'a \pmod{n}, a'b + b' \pmod{n}).$$

Table 1

Scheme	Key Space "K"	Operation "·"
X	$\mathbf{B}_n$	$\oplus$ *(XOR)*
M	$\mathbf{Z}_n^*$	$\times \pmod{n}$
S	$\mathbf{Z}_n$	$+ \pmod{n}$
P	Π_n	*(composition)*
A	$\mathbf{Z}_n^* \times \mathbf{Z}_n$	· *(defined above)*

[0057] When the key space K of an encoding scheme (X, K, E, D) is a group with some operation "·", the encoding and decoding transformations may be uniquely *determined* by the keys. This happens when the key space K is isomorphic, as a group, to the transformation groups $\mathbf{E} = \{E_k \mid k \in K\}$ and $\mathbf{D} = \{D_k \mid k \in K\}$ formed by the encoding and decoding transformations on the message space X; that is, for any $k,k' \in K$,

$$D_k = (E_k)^{-1} = E_{k^{-1}} \text{ and } E_k \circ E_{k'} = E_{kk'}$$

and

$$E_k = (D_k)^{-1} = D_{k^{-1}} \text{ and } D_k \circ D_{k'} = D_{kk'},$$

where is the composition operator of the transformations, which is defined as, for example,

$$E_k \circ E_{k'}(x) = E_{k'}(E_k(x))$$

for all $x \in X$.

[0058] It can be easily checked that all the schemes given in Table 1 above are key-determined. Key-determined encoding schemes permit a systematic way to transfer the right to decode messages from one key holder to another. With the isomorphism between the key space and the transformation groups, the composition of the decoding transformation with one key k and the encoding transformation with another key k' can then be viewed as the encoding transformation determined by the composed key $k^{1} \cdot k$. Let (X, K, E, D) be a key-determined encoding scheme. Suppose $y = E_k(x)$ is the encoded version of the message $x \in X$ with the key $k \in K$. The right to decode the encoded message of x can be transferred from the key holder of k to the key holder of k' in the two-step algorithm shown in Figure 3.

[0059] First, generate a transfer key $t = k^{1} \cdot k$ (step 310). Then encode the message with the transfer key t according to $y' = E_t(y)$ (step 312).

[0060] The algorithm is correct thanks to the property of the key space being isomorphic to the encoding and decoding transformation groups. The correctness can be verified as follows:

$$\begin{aligned} D_{k'}(y') &= D_{k'}(E_t(y)) \\ &= D_{k'}(E_{k^{-1}.k'}(y)) \\ &= D_{k'}(E_{k'}(E_{k^{-1}}(y))) \\ &= E_{k^{-1}}(y) \\ &= D_k(y) \\ &= D_k(E_k(x)) \\ &= x \end{aligned}$$

[0061] The generality of the algorithm makes it immediate to derive the transference steps for the example schemes set forth above. Referring again to Figure 3, for the XOR Scheme X over $\mathbf{B}_n$, to convert $y = E_k(x)$ to $y' = E_{k'}(x)$, first generate a transfer key $t = k \oplus k'$ (step 310). Then encode the message with the transfer key t according to $y' = y \oplus t$ (step 312).

[0062] For the Multiplicative Scheme M over $\mathbf{Z}_n^*$, to convert $y = E_a(x)$ to $y' = E_a(x)$, first generate a transfer key $t = a'a^{-1}$ (mod n) (step 310). Then encode the message with the transfer key t according to $y' = ty$ (mod n) (step 312).

[0063] For the Shift Scheme S over $\mathbf{Z}_n$, to convert $y = E_b(x)$ to $y' = E_b(x)$, first generate a transfer key $t = b' - b$ (mod n) (step 310). Then encode the message with the transfer key t according to $y' = y + t$ (mod n) (step 312).

[0064] For the Substitution Scheme P over Π_n, to convert $y = E_p(x)$ to $y' = E_p(x)$, first generate a transfer key $t = p^{-1} \circ p'$ (step 310). Then encode the message with the transfer key t according to $y' = t(y)$ (step 312).

[0065] As will be described below, it is also possible to transfer the right to decode in product schemes of not only key-determined encoding but also commuting schemes. In order to define commuting schemes, it is necessary to characterize encoding schemes that are essentially equivalent. Suppose that $S = (X, K, E, D)$ and $S' = (X, K', E', D')$ are two encoding schemes with the same message space X. S is said to be *equivalent* to S', denoted by $S \equiv S'$, if there is a bijective (one-to-one and onto) mapping $h : K \rightarrow K'$ such that for each message $x \in X$ and for each key $k \in K$,

$$E_k(x) = E'_{h(k)}(x)$$

and

$$D_k(x) = D'_{h(k)}(x).$$

Clearly, the scheme equivalence relation $\equiv$ is an equivalence relation; that is, it satisfies that, for any encoding schemes S, S', S'', the following hold: (i) $S \equiv S$; (ii) $S \equiv S'$ implies $S' \equiv S$; and (iii) $S \equiv S'$ and $S' \equiv S''$ imply $S \equiv S''$. Thus, equivalent encoding schemes form an equivalence class in that each scheme in the class provides no more and no less functionality than any others in the class.

[0066] The scheme equivalence relation allows one to characterize encoding schemes in several ways. An encoding scheme S is said to be *idempotent* if $S^2 \equiv S$. Many of the encoding schemes are idempotent, including the XOR, multiplicative, shift, substitution, and affine schemes. If a scheme S is idempotent, then there is no point in using the product scheme S^2, as it requires an extra key but provides no more functionality.

[0067] Another characterization on encoding schemes using the scheme equivalence relation $\equiv$ is that of commuting schemes. Two encoding schemes S and S' are said to *commute* if $S \times S' \equiv S' \times S$. Trivially, any scheme commutes with itself. A not-so-trivial example is that of the multiplicative scheme M and the shift scheme S. To see that they commute, i.e., $M \times S \equiv S \times M$, one can compare the equations

$$E_b(E_a(x)) = ax + b \ (\mathrm{mod}\ n)$$

and

$$E_a(E_b(x)) = ax + ab \ (\mathrm{mod}\ n);$$

and find out that the mapping

$$h{:}K_S \times K_M \rightarrow K_M \times K_S$$

defined by

$$h(b,\ a) = (a,\ a^{-1}b \ (\mathrm{mod}\ n))$$

makes the product $S \times M$ isomorphic to the product $M \times S$.

[0068] Product schemes of key-determined and commuting encoding schemes enjoy a systematic way of transferring the right to decode messages. Let $S_1 \times S_2$ be the product scheme of two key-determined and commuting encoding schemes. Suppose that $h = (h_1, h_2) : K_2 \times K_1 \rightarrow K_1 \times K_2$ is the mapping that makes $S_2 \times S_1$ isomorphic to $S_1 \times S_2$, where $h_1 : K_2 \times K_1 \rightarrow K_1$ and $h_2 : K_2 \times K_1 \rightarrow K_2$. First, observe that the product scheme is also key-determined; the product key space $K_1 \times K_2$ is a group with respect to the operation $*$ defined by

$$(k_1, k_2)*(k_1', k_2') = (k_1 \cdot h_1(k_2, k_1'), h_2(k_2, k_1') \cdot k_2').$$

This is because

$$\begin{aligned} E_{(k_1,k_2)} \circ E_{(k_1',k_2')} &= E_{k_1} \circ E_{k_2} \circ E_{k_1'} \circ E_{k_2'} \\ &= E_{k_1} \circ E_{h_1(k_2,k_1')} \circ E_{h_2(k_2,k_1')} \circ E_{k_2'} \\ &= E_{k_1 \cdot h_1(k_2,k_1')} \circ E_{h_2(k_2,k_1') \cdot k_2'} \\ &= E_{(k_1,k_2)*(k_1',k_2')} \end{aligned}$$

[0069] Now, the right to decode the encoded message of x can be transferred from the key holder of k to the key holder of another key k' in the two-step algorithm shown in Figure 3. First, generate a transfer key $t = (h_1(k_2^{-1},k_1^{-1}\cdot k'_1), h_2(k_2^{-1},k_1^{-1}\cdot k)\cdot k'_2)$ (step 310). Then encode the message with the transfer key t according to $y' = E_t(y)$ (step 312).

[0070] The correctness of the transference algorithm is verified by the following equality:

$$\begin{aligned}
E_t(y) &= E_{h_1(k_2^{-1},k_1^{-1}\cdot k_1')} \circ E_{h_2(k_2^{-1}, k_1^{-1}\cdot k_1')\cdot k_2'}(y) \\
&= E_{h_1(k_2^{-1},k_1^{-1}\cdot k_1')} \circ E_{h_2(k_2^{-1}, k_1^{-1}\cdot k_1')} \circ E_{k_2'}(y) \\
&= E_{k_2^{-1}} \circ E_{k_1^{-1}\cdot k_1'} \circ E_{k_2'}(y) \\
&= E_{k_2^{-1}} \circ E_{k_1^{-1}} \circ E_{k_1'} \circ E_{k_2'}(y) \\
&= D_{k_2} \circ D_{k_1} \circ E_{k_1'} \circ E_{k_2'}(y) \\
&= E_{k_1'} \circ E_{k_2'}(x) \\
&= E_{(k_1',k_2')}(x)
\end{aligned}$$

where the last entity can be readily decoded using the key $k' = (k'_1,k'_2)$.

[0071] The method is best illustrated with the following example, applying the affine cipher A over Z_n. Since $A = M \times S$, and M and S are key-determined, commuting schemes, the method described above applies to the affine scheme. As seen before, it is the mapping $h(b, a) = (a, ab)$ that makes $S \times M$ isomorphic to $M \times S$. Thus, $h_1(b, a) = a$ and $h_2(a, b) = ab$ (mod n). The transfer key t from (a, b) to (a', b') can be derived as

$$\begin{aligned}
t &= (h_1(b^{-1},a^{-1}\cdot a'),h_2(b^{-1},a^{-1}\cdot a')\cdot b') \\
&= (a'\cdot a^{-1},h_2(b^{-1},a^{-1}\cdot a')+b') \\
&= (a'\cdot a^{-1},(a'\cdot a^{-1})b^{-1}+b') \\
&= (a'a^{-1},-a'a^{-1}b+b')
\end{aligned}$$

Then, to decode y using a second key (a', b'), first generate a transfer key $t = (a'a^{-1}$ (mod n),$-a'a^{-1}b + b'$(mod n)):$(t_1, t_{2,})$ (step 310). Then encode the message using the transfer key t according to $y' = t_1y + t2$ (mod n) (step 312).

[0072] The methods presented herein for transferring the right to decode messages are *transitive*. This means that two sequential transfers from Alice to Bob and then from Bob to Carol are equivalent to a direct transfer from Alice to Carol. It is important to note that, in each of the example schemes, a transfer key is also a key of the scheme.

[0073] Accordingly, two transfer keys used in the two sequential transfers can be combined to form a transfer key for the direct transfer. Take the affine scheme as an example. Let $k = (a, b)$, $k' = (a', b')$, and $k'' = (a'', b'')$ be the keys for Alice, Bob, and Carol, respectively. Then the transfer keys are $t = (a'a^{-1},-a'a^{-1}b + b')$ from Alice to Bob, $t' =(a''a'^{-1}, -a''a'^{-1}b'+b'')$ from Bob to Carol, and $t'' = (a''a^{-1},-a''a^{-1}b+b'')$ from Alice to Carol. It is straightforward to verify that the composition of t and t' as
keys in the affine scheme yields t'':

$$\begin{aligned}
t\cdot t' &= (t_1't_1,t_1't_2+t_2') \\
&= ((a''a'^{-1})(a'a^{-1}),(a''a'^{-1})(-a'a^{-1}b+b')+(-a''a'^{-1}b'+b'')) \\
&= (a''a^{-1},-a''a^{-1}b+b'') \\
&= t''
\end{aligned}$$

In other words, the composition of sequential transfers of the right to decode messages is memory-less; all the intermediate transfers will not be reflected in the overall transfer.

[0074] It should be noted also that, for the schemes X, M, and S, the transfer key generation step is equivalent to "decoding" k' with k. Thus, the computation needed in the transfer is the same as the one used in the decoding-and-reencoding method for these schemes. One may think that the new method shows no improvement in this efficiency regard, but it has been found that the transfer key is message-independent and hence needs to be computed only once. When the number of messages m involved in the transfer increases, this feature will cut the computation required by the re-encoding method by half. Moreover, the transfer key *t* does not leak any useful information on the keys *k* and *k'*, and a transfer performed according to the methods set forth herein will not reveal the message *x*. These properties make the proposed method appealing when the security of the message *x* and the decoding keys *k* and *k'* is an issue during a transfer.

[0075] A typical system configuration capable of carrying out the methods described with reference to Figure 3 (and described in further detail below) is shown in Figure 4. There are three relevant parties in most proxy encryption applications. An Encryptor 410, a Grantor *A* 412, and a Grantee *B* 414. As will be recognized, the encryption, decryption, and other processing operations performed in the invention are facilitated by a processor (416, 418, 420) under each party's control. Each processor is equipped with memory (422, 424, 426) for data storage and a communication interface (428, 430, 432), capable of sending and receiving messages.

Proxy Encryption Schemes

[0076] The rest of the disclosure, directed to more formal proxy encryption schemes, rather than encoding schemes, is organized as follows. First, a generic proxy encryption scheme is described and characterized according to several criteria. The several following paragraphs fix set forth notation that will be used throughout the disclosure and recall the ElGamal public-key encryption scheme. For the purpose of comparison, this disclosure then lists two existing proxy encryption schemes and examines their properties in comparison to the present invention. Details on the two new proxy encryption schemes are then introduced, together with their security and performance analysis. Applications of these new schemes to massive document distribution and file protection are given thereafter.

[0077] As indicated in the introduction, the goal of proxy encryption is to delegate the decryption right from one to another in secure and efficient ways. For the discussion that follows, it is convenient to define the roles of parties that may be involved in proxy encryption. Two most important roles are those of grantor and grantee. A *grantor* is an original key holder of encrypted messages who wants to delegate the decryption right to someone else. A *grantee* is a key holder designated to perform decryption on behalf of a grantor and thus act as grantor's decryption proxy. In the motivating example in the introduction, Alice is the grantor while Bob is the grantee. Other roles may include an *encryptor* who is an one that originally encrypts messages for the grantor, and a *facilitator* who may help to perform some message processing tasks, such as transforming messages encrypted for the grantor into messages encrypted for the grantee. Certainly, it is not necessary that all these roles are played by different parties. For example, a party may play roles of the grantor and facilitator, as in the Mambo and Okamoto schemes discussed below.

[0078] With these roles in place, a proxy encryption scheme is just a description of how a grantee, possibly with some aid from a facilitator, delegates a grantee the right to decrypt messages originally generated by an encryptor for the grantee. A proxy encryption scheme may consist of four generic steps: message encryption, proxy key generation, proxy transformation and message decryption. These steps will be described in further detail below, with reference to Figure 5.

1. Message encryption E: The encryptor generates an encrypted message using grantor's encryption key and delivers it to the grantor (step 510).
2. Proxy generation π: To delegate the decryption right to the grantee, the grantor generates a proxy key π as a commitment token that allows the grantee to decrypt the message encrypted for the grantor (step 512).
3. Proxy transformation Π: When necessary, the facilitator performs a proxy transformation Π, possibly using the proxy key π, to convert the message encrypted for the grantor to a message encrypted for the grantee (step 514).
4. Message decryption D: Upon receiving the transformed message and possibly the proxy key π, the grantee decrypts the message (step 516).

[0079] Accordingly, it should be observed that the generic scheme above covers the two straightforward solutions to proxy encryption mentioned in the introduction. The re-encryption scheme is a special case where the grantor (Alice) is also the facilitator who actually decrypts the message and then encrypts for the grantee (Bob), and the proxy π can be considered as a collection of grantor's decryption key and grantee's encryption key, which is used only by the grantor and not by the grantee. The scheme of passing grantor's decryption key to the grantee is another special case of the generic scheme, where the proxy key is the decryption key and the proxy transformation is the identity transformation.

[0080] However, not all schemes that can be derived from the generic one above are qualified as proxy encryption schemes. Intuitively, a proxy encryption scheme has to satisfy some basic requirements, namely delegation, security, transitivity and performance, as described below.

Delegation. To ensure that, at the end of the message decryption step, the grantee is able to recover the original message correctly, the following equation must hold for any message *m*:

$$D(\Pi(E(m,e_A),\pi),d_B,\pi) = m,$$

where $E(m,e)$ is an encryption function of message m under encryption key e, $D(c,d,\pi)$ is a corresponding decryption function of encrypted message c under decryption key d and possibly proxy key π, $\Pi(c,\pi)$ is the proxy function that converts encrypted message c according to proxy key π, and e_A, e_B, d_A, and d_B are the encryption and decryption keys of the grantor A and grantee B, respectively.

In addition to the correctness above, the functionality of delegation should be guaranteed. In one form, this means that, after the proxy key is issued and the proxy transformation is completed, the message decryption step should require no private information from the grantor, and it should be carried out solely by the grantee. In another form, this is equivalent to undeniability of the delegation from the grantor; that is, once the proxy key is created and proxy transformation is performed, the grantor should not be able to deny the delegation, without seeking other means such as preventing the grantee from obtaining the proxy key and receiving the transformed message. As a consequence of this functionality, the grantor's decryption key can be destroyed with grantee's decryption key and possibly the proxy key maintaining the ability to decrypt the message. (This is useful in the file protection application later in Section 6.)

Security. In essence, a proxy encryption scheme is also an encryption scheme at least from the grantee's point of view. The introduction of proxy keys and transformations must in no way com-promise security and privacy of the encryption. Thus, it should be at least computationally hard for any unauthorized third party to recover the original message and decryption keys of the grantor and grantee from publicly available information. Moreover, forging valid proxy keys by any untrusted party should be very hard. It must be clear, though, that generating the proxy key π requires knowledge of at least the decryption key of the grantor; otherwise the underlying encryption system is not secure.

Transitivity. Naturally, the proxy relationship should be transitive. After the grantor delegates the decryption right, the grantee should be able to act as a new grantor to delegate the right further to another grantee, by just following the same scheme. Moreover, it should be possible for someone, say the first grantor, to delegate the right directly to a new grantee by combining all intermediate proxy keys into one proxy key and composing all consecutive proxy transformations into one transformation.

Performance. As the re-encryption scheme is an intuitive, straightforward solution to proxy encryption and it satisfies the above delegation, security and transitivity requirements, any practically useful proxy encryption scheme should have no degradation in computational performance when compared with the re-encryption scheme.

Proxy encryption schemes may vary according to their application requirements. They can be categorized according to many aspects. Obvious ones include whether they are public-key or private-key based, and whether their security measures are perfect in the information theoretical sense or rely on intractability of some computational problems. The following aspects ones are related to the proxy key and transformation.

Confidentiality. While secrecy of messages and decryption keys has to be enforced, secrecy of proxy keys and proxy transformations may not be a mandatory requirement. A scheme is called public if proxy keys it generates may be published without compromising its security and proxy transformations applied in untrusted environments; otherwise, the scheme is private. In a private scheme, when a proxy key is transferred from the grantor to the facilitator and grantee, care must be taken to protect the proxy key from disclosure. As a result, the proxy transformation which uses the proxy key must be performed in private as well.

Commutativity. In terms of messages, the grantee must be unconditionally trusted by the grantor, since proxy encryption by definition allows the former to decrypt on behalf of the latter. However, the trust model may be different for their private information. A proxy encryption scheme is commutative if the grantor and grantee have to trust each other with regard to their private keys; otherwise, it is non-commutative. A commutative example is that the proxy key is such created that either one of the grantor and grantee can obtain other's decryption key from it. Whenever this is the case, the proxy encryption mechanism may be simplified to a key exchange protocol that allows the grantee to use grantor's decryption key to decrypt the encrypted messages directly.

Generality. In many cases, the grantor wants to restrict the scope of the delegated decryption right. Often intended restrictions include that the proxy key may only be used by a designated grantee, that the proxy key may only be applicable to a specific message, or that the proxy transformation may only be applied by a specific facilitator. For example, when a proxy encryption scheme is used in some applications like key escrow, it would be ideal that proxy keys are independent of messages they will apply to. But for occasional delegation such as securely specifying inheritance in someone's will, it may be highly desirable that a proxy key can only be restricted to a designated party (e.g., a grandchild), applicable to a specific message (e.g., some portion of the will) and possibly used in the proxy transformation by a particular party (an attorney).

Degenerateness. When used in the extreme situation where the grantor and grantee are a same person with a same decryption key, a proxy encryption scheme should reduce to a regular encryption scheme, without introducing any complications (such as non-trivial proxy keys and transformations, and the requirement of an extra facilitator).

[0081] As will be shown below, the Mambo and Okamoto schemes are private and non-commutative. Proxy keys in their schemes can be either message-independent or dependent but are not restricted to designated grantees. The Blaze and Strauss scheme is just opposite: it is public but commutative, and its proxy keys are message-independent but uniquely associated with designated grantees. In comparison, the schemes according to the invention set forth herein are public and non-commutative, and their proxy keys are message-dependent and restricted to designated grantees.

Proxy Encryption Using the ElGamal Cryptosystem

[0082] As the proxy encryption schemes discussed below in this disclosure will all be based on discrete logarithms in multiplicative groups, a formal setting which is common to all these encryption schemes is hereby adopted. The notation used herein recalls the ElGamal encryption scheme. Encryption schemes based on discrete logarithms are particularly advantageous because of their technical advantages over RSA-type schemes and their natural generalizations to many finite groups such as elliptic curve groups over finite fields.

[0083] As set forth above, for any natural number n, let $\mathbf{Z}_n = \{0,1,...,n-1\}$ denote the ring of integers modulo n, and let $\mathbf{Z}_n^* = \{m \in \mathbf{Z}_n \mid \gcd(m,n) = 1\}$ denote the multiplicative group of $\mathbf{Z}_n$. Note that, when n is a prime, $\mathbf{Z}_n^* = \{1,...,n-1\}$. For a modulo n and a number a that is relatively prime to n, let a^{-1} denote the multiplicative inverse of a modulo n; that is, a^{-1} is the element that satisfies $aa^{-1} \equiv 1 (\mathrm{mod}\ n)$.

[0084] An element a of $\mathbf{Z}_p^*$ is said to be of order m if the number of its powers modulo n is m. A generator g of $\mathbf{Z}_n^*$, if it exists, is an element of order $| \mathbf{Z}_n^* |$ (the size of $\mathbf{Z}_n^*$); in this case, $\mathbf{Z}_n^*$ is a cyclic group. When n is a prime, every element of $\mathbf{Z}_n^*$ except 1 is a generator of $\mathbf{Z}_n^*$.

[0085] Let $\mathbf{Z}_n^*$ be a cyclic group with a generator g. The discrete logarithm of an element x to the base g, denoted as $\log_g x$, is the unique integer a, $0 \le a \le n - 1$, such that $x = g^a (\mathrm{mod}\ n)$. The *discrete logarithm problem* is that, given a prime p, a generator g of $\mathbf{Z}_p^*$, and an element $x \in \mathbf{Z}_p^*$, find the integer a, $0 \le a \le p\text{-}2$, such that $g^a \equiv x (\mathrm{mod}\ p)$.

[0086] A very closely related problem is the *Diffie-Hellman problem:* given a prime p, a generator g of $\mathbf{Z}_p^*$, and elements $g^a (\mathrm{mod}\ p)$ and $g^b (\mathrm{mod}\ p)$, find $g^{ab} (\mathrm{mod}\ p)$. The discrete-logarithm problem is at least as hard as the Diffie-Hellman problem, because any solution to the former problem can be used to solve the latter problem.

[0087] The ElGamal encryption scheme shown in Figure 6 is a part of a discrete-logarithm based, public-key cryptosystem proposed by ElGamal for both encryption and digital signature. See T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithm," *IEEE Trans. on Information Theory,* Vol. 31, pp. 465-472 (1985).

[0088] Referring now to Figure 6 in detail, the ElGamal scheme is set up (step 610) by establishing two public parameters p and g, where p is a prime (typically 512 bits in length), such that p-1 has a large (typically 160 bit) prime factor q (e.g., $p=2q+1$) and g is a generator in $\mathbf{Z}_p^*$. A private key for a user is set (step 612) by uniformly choosing a random number $a \in \mathbf{Z}_{p-1}^*$. Its related public key is calculated (step 614) as $\alpha = g^a (\mathrm{mod}\ p)$. The user publishes α and keeps a secret.

[0089] To encrypt a message m to be sent to user A with public key α, a random number $k \in \mathbf{Z}_{p-1}^*$ is uniformly chosen (step 616), and a pair of numbers (r,s), together representing the encrypted message to be sent to A, is calculated (step 618) as follows:

$$r = g^k (\mathrm{mod}\ p) \text{ and } s = m\alpha^k (\mathrm{mod}\ p).$$

[0090] To decrypt the message (r,s), the recipient A recovers the message m (step 620) by calculating

$$m = s(r^a)^{-1} (\mathrm{mod}\ p).$$

[0091] Note that the selection of the public parameters is intended to establish equation $g^{p-1} (\mathrm{mod}\ p) \equiv 1$ (Fermat's little theorem). These parameters should be authentically known to all users. They can be chosen, say, by some trusted authority. Also, the way that private key a is chosen ensures that the inverse a^{-1} of a modulo p-1 exists and is unique.

[0092] Unlike the RSA public-key encryption scheme, the ElGamal scheme is non-deterministic, since the encrypted message also depends on the random number k. Indeed, it is similar in nature to the Diffie-Hellman key exchange protocol; the key established between the sender and receiver for encrypting and decrypting the message m is g^{ak} (mod p) from $r = g^k (\mathrm{mod}\ p)$ (part of the encrypted message) and $\alpha = g^a (\mathrm{mod}\ p)$ (the public key of A). Nevertheless, the security of the ElGamal encryption scheme relies on the intractability of the discrete logarithm problem and the Diffie-Hellman problem. To date, practice in seeking optimal algorithms for the discrete logarithm problem has not found any